Name:	
ID:	

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Quiz 1E **Spring 2020, CU-Boulder**

Instructions: This quiz is closed book, closed book, and an individual effort. Electronic devices (including, but not limited to phones, smart watches, FitBits, calculators, etc.) are NOT allowed on your person (including in your pocket). Possession of such electronics is grounds to receive a 0 on this quiz. Proofs should be written in **complete sentences. Show all work to receive full credit.**

Please provide these:
Left neighbor name :
Right neighbor name :

Name:

ID:

CSCI 3104, Algorithms **Profs. Chen & Grochow**

Quiz 1E **Spring 2020, CU-Boulder**

1. *For the following algorithm, clearly state a* ***useful*** *loop invariant. Note that you are being asked to provide a* ***loop invariant ONLY****, and* ***NOT*** *a loop invariant proof.*

```
CountMultiplesOf5(A[1, ..., n]) : //array A is not empty
    count = 0
    for i = 1 to length(A) {
        if A[i] mod 5 == 0{
            count = count + 1
    }}
    return count
```

Solution: Prior to the start of iteration i, `count` stores the number of elements in $A[1, ..., i-1]$ that are multiples of 5.

2. *Consider the following algorithm.*

```
moveMinToFront(A[1, ..., n]):
   for i = n; i > 1; i--{
       if A[i] < A[i-1]{
          swap(A[i], A[i-1])
   }}
```

We have the following loop invariant: At the beginning of each iteration i, $A[i]$ is the smallest element of subarray $A[i, ..., n]$.

Your job is to provide the components for the loop invariant proof in the parts below.

(a) Using the loop invariant above, provide the **initialization** component of the loop invariant proof. That is, show that the loop invariant holds before the first iteration of the loop is entered.

Solution: Prior to the start of the loop, we consider the sub-array $A[n, \dots, n]$, which has a single element: $A[n]$. Since $A[n, \dots, n]$ has only a single element, that element is trivially the smallest element in that subarray. So the loop invariant holds prior to the start of the loop.

(b) Using the loop invariant above, provide the **maintenance** component of the loop invariant proof. That is, assume the loop invariant holds just before the i-th iteration of the loop, and use this assumption to show that it still holds just before the next iteration.

Solution: Suppose that the loop invariant holds prior to the start of iteration i. We show that loop invariant holds prior to the start of the next ($(i-1)$st) iteration. As the loop invariant holds prior to the start of iteration i, $A[i]$ is the smallest element in $A[i, \ldots, n]$. During iteration i, the algorithm compares $A[i]$ and $A[i-1]$. If $A[i-1] \leq A[i]$, the algorithm takes no further action in the ith iteration. We note that in this case that $A[i-1]$ is the smallest element in $A[i-1, \ldots, n]$, and so the loop invariant holds prior to the start of the next iteration.

Alternatively, suppose $A[i] < A[i-1]$. Then the algorithm swaps $A[i]$ and $A[i-1]$. So $A[i-1]$ is the smallest element in $A[i-1, \ldots, n]$, and so the loop invariant holds prior to the start of the next iteration.

(c) Using the loop invariant above, provide the **termination** component of the loop invariant proof. That is, assume the loop invariant holds just before the last iteration. Then argue that the loop invariant holds after the loop terminates, based on what happens in the last iteration of the loop. Finally, use this to argue that the algorithm overall is correct.

Solution: Suppose the loop invariant holds prior to the start of the last iteration. So $A[2]$ is the smallest element in $A[2, \ldots, n]$. During the last iteration of the loop, the algorithm compares $A[2]$ and $A[1]$. If $A[1] \leq A[2]$, the algorithm takes no further action. We note that in this case that $A[1]$ is the smallest element in A, and so the loop invariant holds.

Alternatively, suppose $A[2] < A[1]$. Then the algorithm swaps $A[2]$ and $A[1]$. So $A[1]$ is the smallest element in A, and so the loop invariant holds.

As $A[1]$ is the smallest element in A after the loop terminates, we have that the algorithm is correct. The result follows.